AI Enabled Car Parking Using OpenCV

Team Members:

V M S VAMSI KRISHNA ALAPATI (20481A5459) (Team Lead)

TAGIRISA TEJASRI (20481A5453)

GARIKIPATI LOHITH (20485A5417)

V H NAGA SAISRI (20481A5460)

Table of Contents

1. INTRODUCTION

1.1 Overview: -

It is a hectic task to park the car in a parking lot. This task requires to look for the availability of slots and park as fast as a person can before others park. So, this AI enabled car parking system aims to develop a working system that utilizes OpenCV and computer vision techniques to optimize and automate the car parking process. The intelligent car parking system we proposed uses Hough Transform(CTC), OpenCV, Video-Image processing, Circle detection for our fast rush growing urban areas. Monitoring and management of traffic parameters is a recent trend in research development.

Here we focus on parking component of traffic parameters. While parking a car can burn more fuel and also take a lot of time. Our system analyses and detects the car in parking lot using cameras and other imaging devices. It also provides real-time information about available slots to park the cars to the drivers. OpenCV is utilized for this purpose, which is an open-source computer vision library. To capture real-time video footage of the parking lot, cameras are used. And then applies computer vision algorithms for the detection and tracking vehicles.

This project involves the following steps:

1. Video capture: Cameras installed in the parking lot captures video streams.

2. Vehicle Detection: To identify the vehicles in the video stream based on size, shape, and other visual features, the computer vision techniques are used.

3. Video Tracking: When a vehicle is detected, our system tracks its movement across subsequent video frames to determine its trajectory.

4. Parking Space Monitoring: To identify the available spaces and track their occupancy status our system analyses the parking spaces.

5. Parking Guidance: The system guides the incoming vehicles to available parking spaces based on the occupancy status of parking spaces.

6. User Interface: To display the real-time parking lot footage, occupancy status, and guidance information to the users, a user interface is provided to enable the user interact the system.

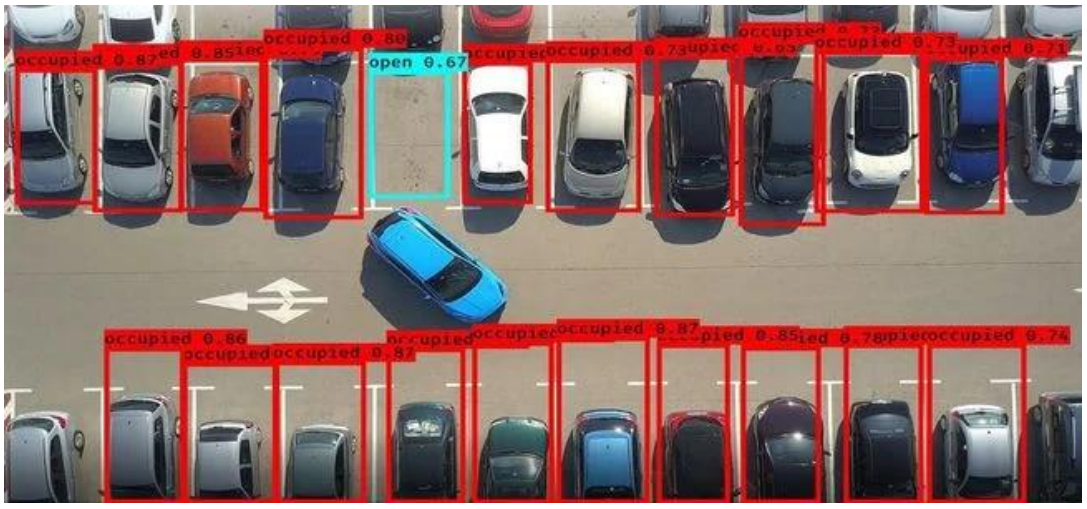
1.2 Purpose: -

The purpose of the "AI Enabled car parking using OpenCV (Open Source Computer Vision)" project documentation is to automate and optimize the process of parking vehicles in a parking lot or a parking garage. OpenCV is a popular library for computer vision and image processing, and when combined with artificial intelligence (AI) techniques, it can offer numerous benefits for efficient car parking management. AI algorithms can be trained to recognize and detect vehicles, parking spaces, and obstacles, allowing the system to autonomously guide a vehicle into a parking spot without human intervention.

This documentation aims to convey a clear understanding of the project's scope, objectives, and significance. It provides context regarding AI-powered parking systems can help reduce the time taken for drivers to find parking spots, as they can quickly identify available spaces and guide the driver to the closest one.

With efficiently utilizing parking space and reducing the time spent searching for parking spots, AI-enabled parking systems can potentially lower operational costs and increase revenue for parking lot operators by the integration of Flask-based web application, this documentation elucidates the purpose, functionality, and user experience provided by the application. It explains how users can interact with the model through the web interface to detect the empty spots in the parking lot or garage.

AI-enabled car parking using OpenCV offers a smart and efficient solution for managing parking spaces, enhancing the overall parking experience, and contributing to smarter and more sustainable urban environments.



2. LITERATURE SURVEY

2.1 Existing problem: -

For AI Enabled Car Parking Using OpenCV, we have created and implemented a framework. The experimental results demonstrated the suggested system's ability to provide accurate data. The issue of parking in crowded regions has been addressed using a variety of strategies and forms. An approach for counting the cars at the checkpoint from which the variety of open parking spots can be counted was once proposed via Ming- Yee Chiu et al. Induction loop sensors are positioned below the road surface to elevate out the counting.

Although the usage of sensors was once much less highly-priced, they aren't fluently told by environmental factors, and they reliably detect, their installation was challenging and resulted in road damage. In the event of a problem, maintenance was extremely grueling . Grounded on vision- based techniques, various categories of discovery styles are described and it can be a crucial and challenging task.

The accuracy of car detection and parking guidance heavily relies on the performance of computer vision algorithms. Variations in lighting conditions, weather, and occlusions can impact the system's ability to accurately detect cars and parking spaces. The existing problem revolves around the lack of an efficient and data-driven solution for predictingparking lots with high accuracy.

Real-time monitoring and analytics in the context of AI-enabled car parking using OpenCV refer to the continuous and immediate tracking of parking lot data and the analysis of this data to gain valuable insights. This capability allows parking operators and city planners to make data-driven decisions, optimize parking operations, and improve overall parking management. It plays a vital role in optimizing parking operations, enhancing user experience, and contributing to smarter and more efficient parking management in modern urban environments.

The "AI Enabled car parking using OpenCV" project seeks to address these challenges, specifically OpenCV. This model aims to accurately estimate parking slots based on a comprehensive datasetof property features. Additionally, the project includes the development of a Flask-based web application, which offers a user-friendly interface for interacting with the predictive model.

2.2 Proposed solution: -

This project aims to empowers that entire parking lot that's open for parking can be analyzed by the camera using vision- grounded ways; the facts is also analyzed, and the output will specify the unique quantity and regions of the open parking spaces. The solution can enhance the efficiency of detecting the parking lots in the parking garage. Ultimately, the project aims to detect the empty slots in the parking lot revolutionizing the way detecting by the rectangular lanes and enabling automatically detecting the empty slots.

The vision-based framework recognizes the free parking spot by utilizing cameras around the vehicle and distinguishes the items in parking spot by ascertaining the profundity and separation of the question from the vehicle and divider edges. This data sends to the innovation that helped with ADAS framework.

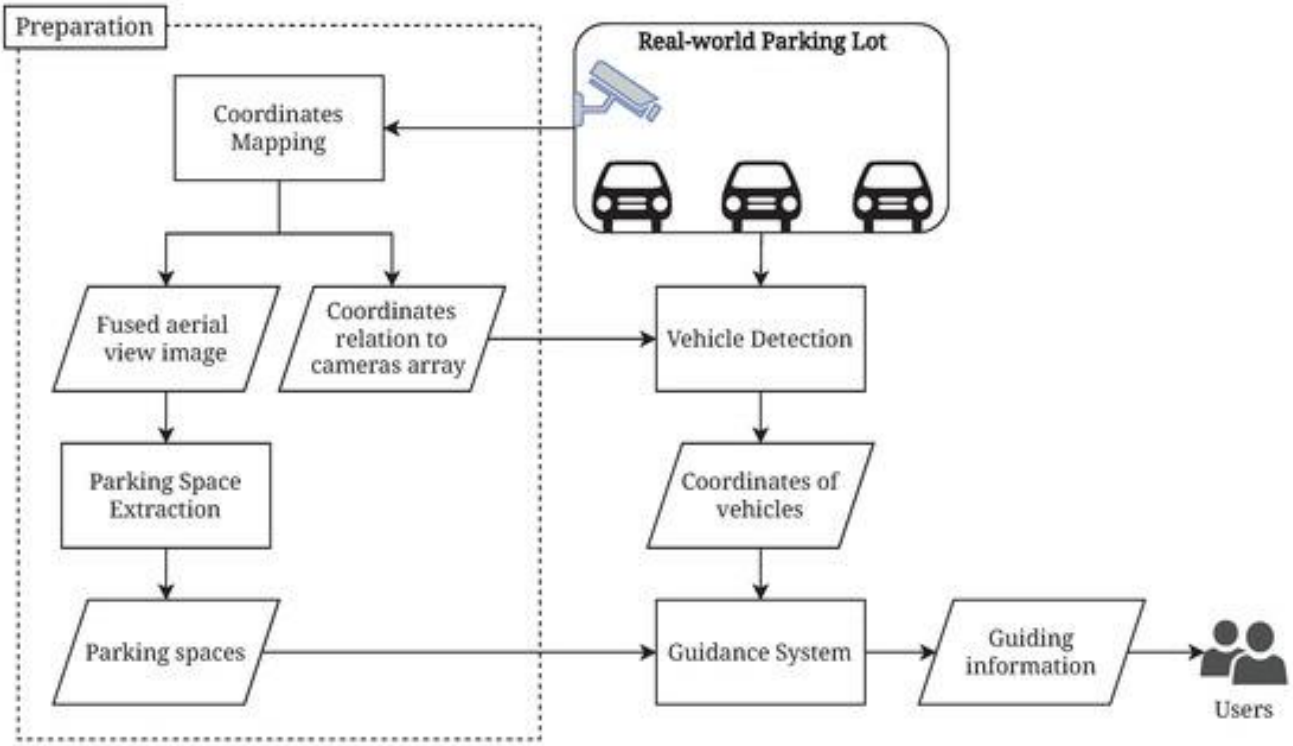
Videos were acquired from the top view of parking arena, heighted camera. To strengthen the recognition capacity of system video data was captured at different environmental conditions and temporal shifts. Video is segmented into frames. Then from each segment a key frame is extracted and further processing is applied on this key frame, to reduce computational complexity. When car enter or leave the parking lot from parking arena, motion of car is estimated.

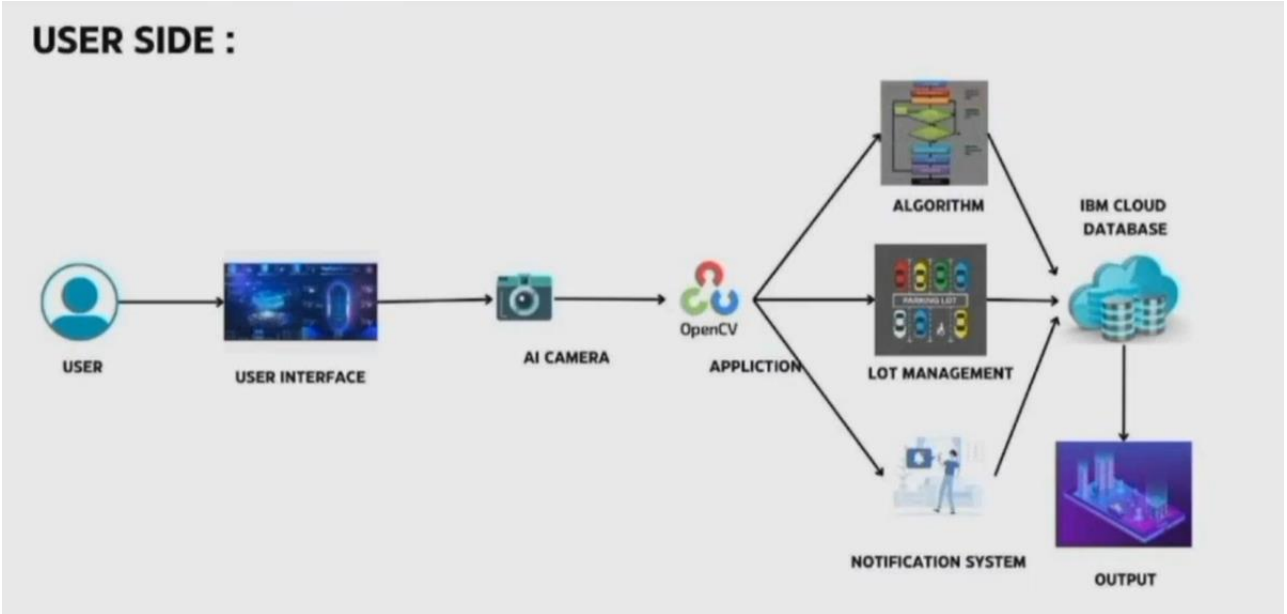
The predictive model generates valuable insights into rental price trends and patterns. Real estate market analysts can leverage this information to study the dynamics of rental prices across different locations, property types, and market conditions. This knowledge can assist in strategic decision-makingand investment planning.

The user-friendly Flask-based web application streamlines the process of obtaining will automatically divide the parking area in multiple blocks by drawing virtual lines on the parking lot and efficiently detect the cars in these virtual parking blocks. The accuracy of our proposed system is better than the Color based twin ROI detection technique and edge-based detection technique used in existing parking systems and it is developed in python.

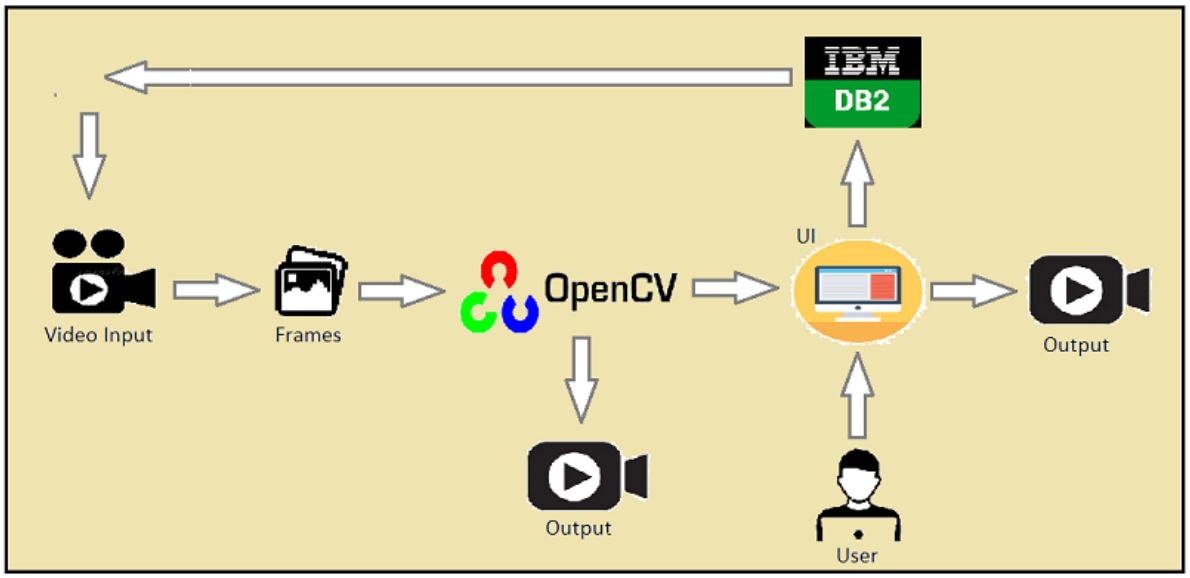
3. THEORETICAL ANALYSIS

3.1 Block diagram: -





3.2 Software designing: -



- **Web Application: -**

A Flask web app is a web application built using the Flask framework, which is a lightweight and easy- to-use Python web framework. Flask allows developers to create web applications quickly and efficiently by providing essential tools and utilities. It follows the WSGI (Web Server Gateway Interface) standard,making it compatible with various web servers.

- **Frames: -**

An image is defined as a two-dimensional function, F(x,y) where x and y are spatial coordinates, and the amplitude of F at any pair of coordinates (x,y) is called the intensity of that image at that point. When x,y, and amplitude values of F are finite, we call it a digital image.

- **OpenCV: -**

The Image processing library mainly focused on real-time computer vision with application in wide-range of areas like 2D and 3D feature toolkits, facial & gesture recognition, Human-computer interaction, Mobile robotics, Object identification and others.

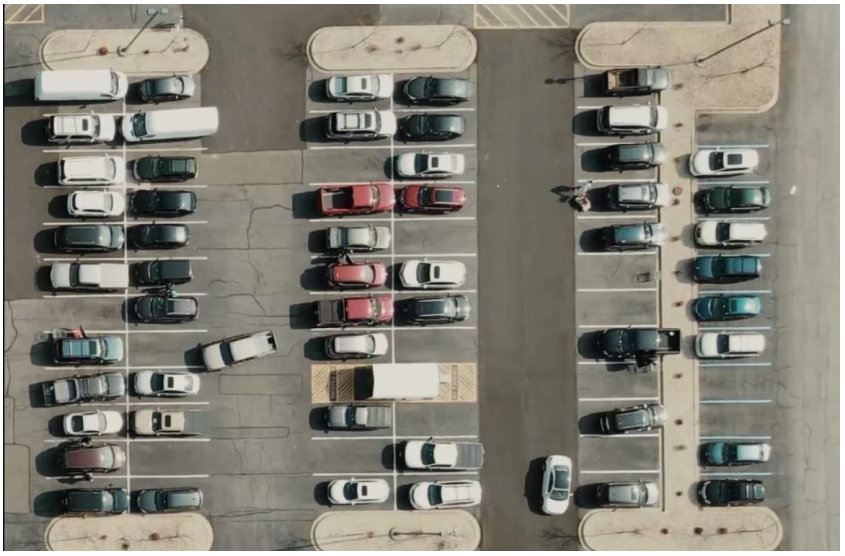
- **IBM DATABASE: -**

IBM Db2 on Cloud is a fully managed cloud-based relational database service offered by IBM. It is designed to provide a scalable, secure, and high-performance database solution in the cloud, allowing businesses to focus on application development and data management.

4. EXPERIMENTAL INVESTIGATIONS

OpenCV is an extensive open source library (available in python, Java, and C++) that's used for image analysis and is pretty neat. The lofty goal for my OpenCV experiment was to take any static image or video of a parking lot and be able to automatically detect whenever a parking space was available or occupied. What I was able accomplish was to detect how many spots were available in a parking lot, with just a bit of upfront work by the user.

Original Image:-



Gray Filter:-

gray=cv.cvtColor(img,cv.COLOR_BGR2GRAY)

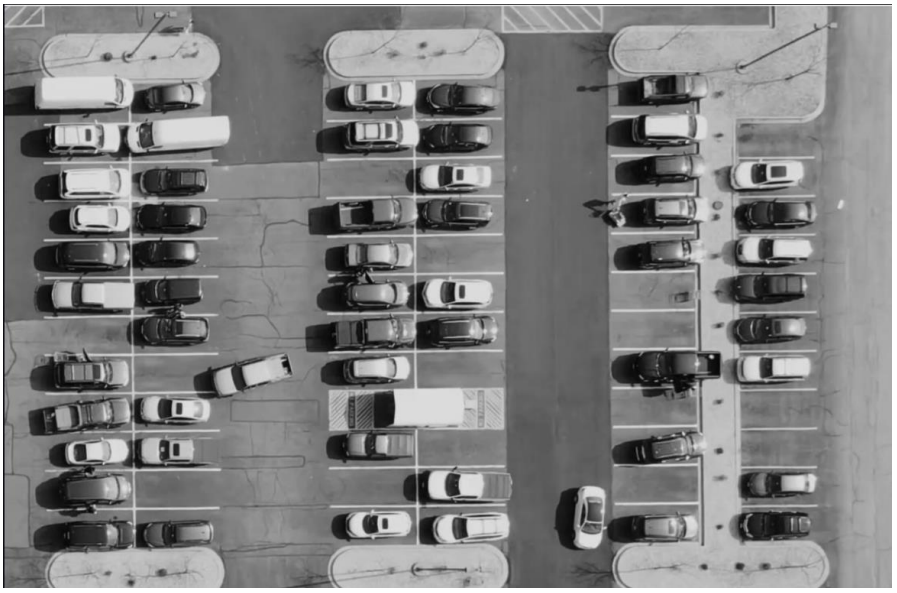
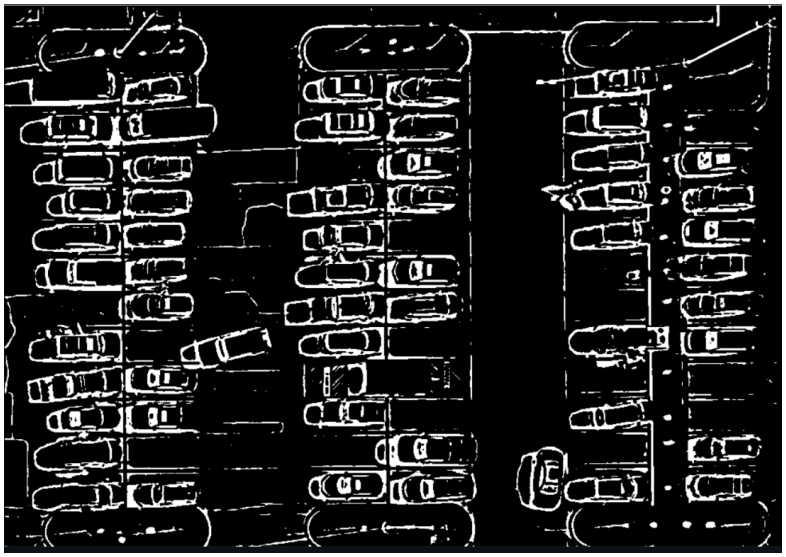


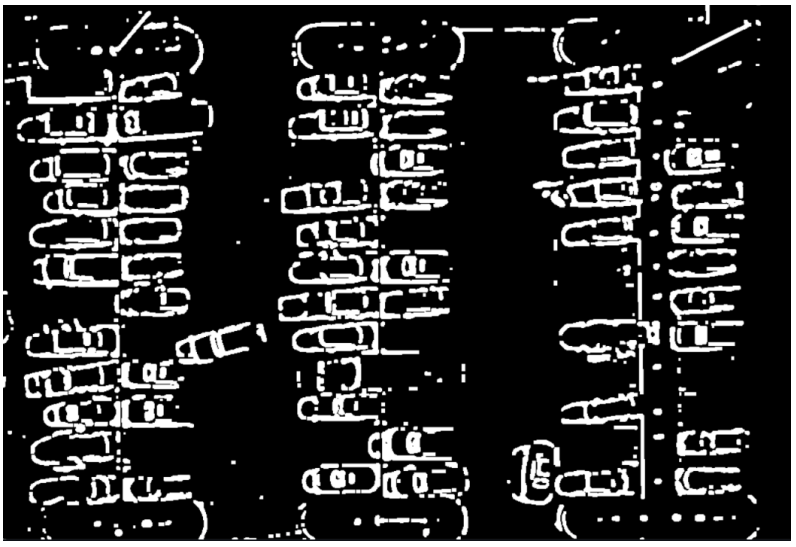
Image Tresholding:-

imgThreshold=cv2.adaptiveThreshold(imgBlur,255,cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV,25, 16)



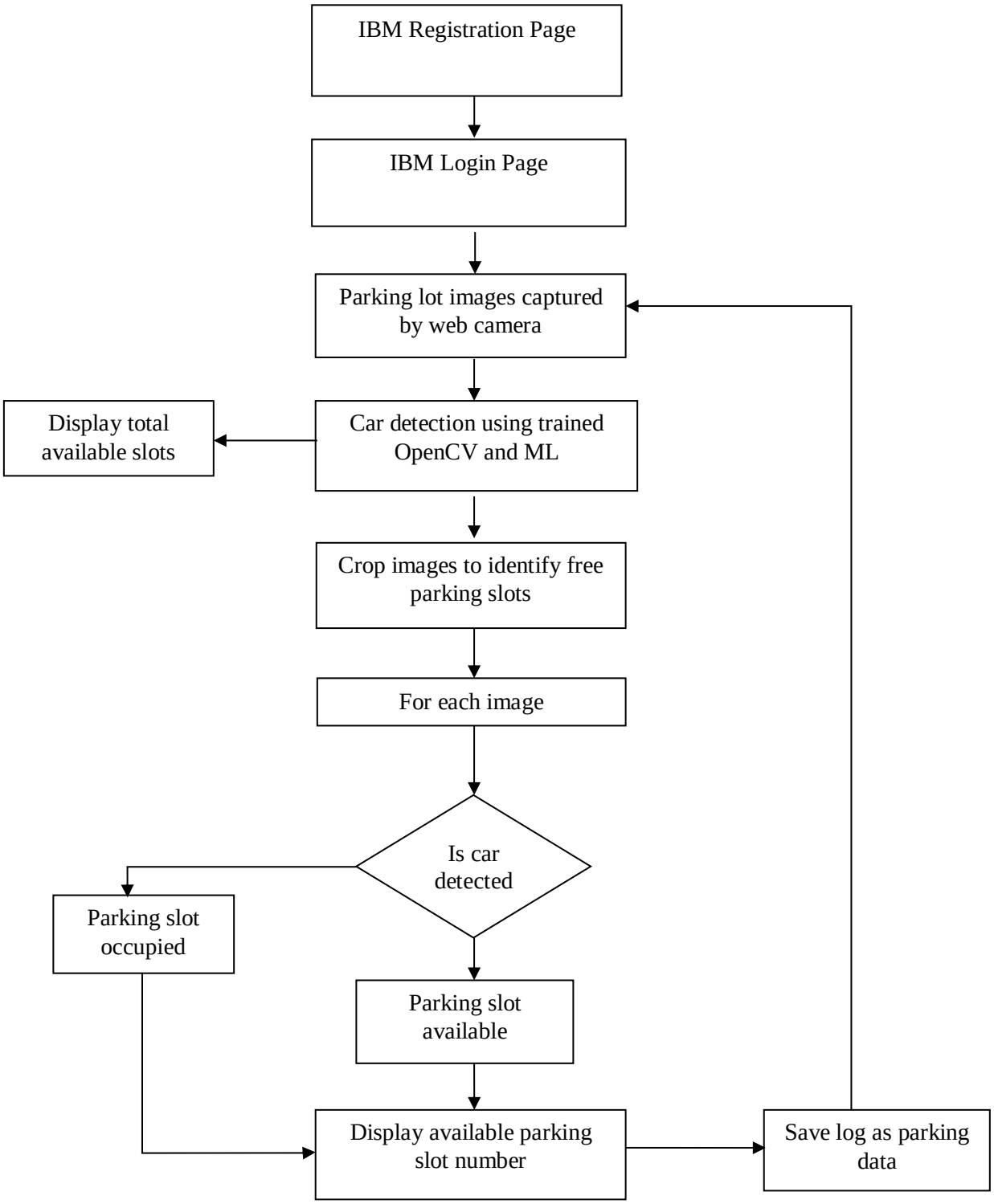
Applying Dilation to image median:-

kernel = np.ones((3, 3), np.uint8) imgDilate =cv2.dilate(imgMedian,kernel,iterations=1)



The experimental investigation aims to provide a comprehensive assessment of the model's performance and the web application's usability. And it can easily identify the parking slots after applying many filters like gaussian blur and median and thresholding the blurred image to remove the obstacles and trees and remove unwanted objects from the image and acquire the slots.

5. FLOWCHART


IBM Registration Page
IBM Login Page
Parking lot images captured by web camera
Display total available slots
Car detection using trained OpenCV and ML
Crop images to identify free parking slots
For each image
Is car detected
Parking slot occupied
Parking slot available
Display available parking slot number
Save log as parking data

6. RESULT

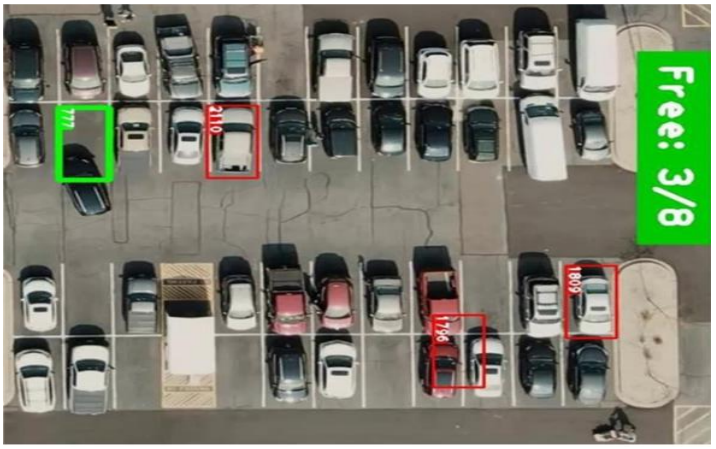
The primary route-way of the proposed algorithm for parking space discovery are

1. The parking lot will be live- streamed by the camera to the system.
2. When a horseless carriage pulls into or out of the parking space, film making are taken.
3. Grayscale images are created by converting RGB images.
4. Make adjustments Choosing the parking lot's equals first is a good idea.

This will remove any unnecessary white space from the image other than the parking lot. • Next, decide where the single parking space's parallels are. As a result, the parking lot will be divided into spaces of cognate size. 5. In order to turn the parking lot into black and the auto into white, each block is first converted from grayscale to double and then to inverse binary. 6.To determine if a block contains a car or not, a threshold value is computed for each block. Blocks are free and available for parking if their value is less than a threshold value, and they're occupied if their worth exceeds the threshold.

The primary objective of this project was to develop a robust model capable of accurately detecting parking spots based on various property images. Through extensive experimentation and evaluation, we are delighted to report that the OpenCV model achieved an impressive accuracy in detecting the available parking spot.

In order to overcome some of the weakness that was observed in the development of this vision based system, improvements can be made in the type of camera used where the web cameras can be tied in together with CCTV cameras. In addition, the coordinate method used in this paper in selecting the specific parking locations has limited the usage by making the each parking bay location fixed and thus limits the camera to be at a fixed location.



7. ADVANTAGES & DISADVANTAGES

Advantages: -

- Our proposed method will automatically divide the parking area in multiple blocks by drawing virtual lines on the parking lot and efficiently detect the cars in these virtual parking blocks.
- The accuracy of our proposed system is better than the Color based twin ROI detection technique and edge-based detection technique used in existing parking systems.
- The proposed method is implemented in MATLAB. The online system is getting images from the camera while offline system is getting images from a video file. The result of the online system shows that the proposed algorithms has efficiently detected the available parking slots and notify the drivers.

Disadvantages:

- Drivers can take more time to find free parking places.
- It requires some staff for choose free parking space.
- There are chances for vehicle vandalism if staff is not available.
- While the model demonstrates exceptional accuracy, it is essential to acknowledge its limitations.

8. APPLICATIONS

The proposed solution has various applications, including:

- AI Enabled Car Parking System is often used in locations where a multi-story parking garage would be too large, too costly or impractical.
- Examples of such applications include, under or inside existing or new structures, between existing structures and in irregularly shaped areas.
- It can also be applied in situations similar to multi-story parking garages such as freestanding above ground, under buildings above grade and under buildings below grade.

9. CONCLUSION

As conclusion, The results obtained from the testing of the vision-based car park system on model cars and car parks, it indicates that the probability of the utilization of a camera or vision-based system to monitor car parks is feasible. This is seen in the results that indicate the accuracy in determining the presence of a car in a parking bay or the information that can be provided on the location of the available parking bays.

The development of the vision-based car park system shows the feasibility of utilizing simple cameras such as web cameras to monitor car parks. However, the utilization of such cameras has indicated some weaknesses in terms of the accuracy of the detection due to the limitations of the camera.

In order to overcome some of the weakness that was observed in the development of this vision-based system, improvements can be made in the type of camera used where the web cameras can be tied in together with CCTV cameras. In addition, the coordinate method used in this paper in selecting the specific parking locations has limited the usage by making each parking bay location fixed and thus limits the camera to be at a fixed location and it used to detect the empty spots in the parking lot.

10. FUTURE SCOPE

Future enhancements for the project include:

- Hook up a webcam to a snort Pi and have live parking monitoring at home
- Alchemize parking lot video to have overview perspective(for clearer globules)In addition, it provides automatic billing, as well as eliminating traffic congestion.
- Utilizing a multilevel parking technique, this work can be further developed into a fully automated system.

11. BIBLIOGRAPHY

[1] Gaikwad, M.J., Asole, P.S. and Bitla, L.S., 2022, January. Effective Study of Machine Learning Algorithms for Heart Disease Prediction. In 2022 2nd International Conference on Power Electronics & IoT Applications in Renewable Energy and its Control (PARC) (pp. 1-6). IEEE.

[2] Sadani, U., Bitla, L., Vairagade, R. and Ghule, V., 2022, April. Power and Delay Efficient Three-Input XOR/XNOR With Systematic Cell Design Methodology. In 2022 10th International Conference on Emerging Trends in Engineering and TechnologySignal and Information Processing (ICETET-SIP-22) (pp. 1-5). IEEE.

[3] Vairagade, R., Bitla, L., Judge, H.H., Dharpude, S.D. and Kekatpure, S.S., 2022, April. Proposal on NFT Minter for Blockchain-based Art-Work Trading System. In 2022 IEEE 11th International Conference on Communication Systems and Network Technologies (CSNT) (pp. 571-576). IEEE.

[4] S. Ma, H. Jiang, M. Han, J. Xie and C. Li, "Research on Automatic Parking Systems Based on Parking SceneRecognition," in IEEE Access, vol. 5, pp. 21901-21917, 2017, doi: 10.1109/ ACCESS.2017.2760201

[5] Keat, C.T.M.; Pradalier, C.; Laugier, C. Vehicle detection and car park mapping using laser scanner. In Proceedings ofthe IEEE/ RSJ International Conference on Intelligent Robots and Systems, Edmonton, AB, Canada, 2–6 August 2005; pp.2054–2060.

APPENDIX

11.1. Source Code

The project's source code is available in the 'app.py' file, containing the Flask web application.

App.py:

```
from flask import Flask, render_template, request, session, redirect, url_for
import cv2
import pickle
import cvzone
import numpy as np
import ibm_db
import re
app = Flask(__name__)
app.secret_key = 'a'
conn = ibm_db.connect("DATABASE=bludb;HOSTNAME=9938aec0-8105-433e-8bf9-0fbb7e483086.c1ogj3sd0tgtu0lqde00.databases.appdomain.cloud;PORT=32459;SECURITY=SSL;SSLServerCertificate=DigiCertGlobalRootCA.crt;UID=zhl76206;PWD=jjKL3036L69a6Jxo;","","")
@app.route('/')
def project():
    return render_template('index.html')
@app.route('/index.html')
def home():
    return render_template('index.html')
@app.route('/contact.html')
def cont():
    return render_template('contact.html')
@app.route('/model')
def model():
    return render_template('model.html')
@app.route('/login.html')
def login():
    return render_template('login.html')
@app.route('/aboutus.html')
def aboutus():
    return render_template('aboutus.html')
@app.route('/signup.html')
def signup():
    return render_template('signup.html')
@app.route("/signup", methods=['POST', 'GET'])
def signup1():
    msg = ''
    if request.method == 'POST':
        name = request.form["name"]
        email = request.form["email"]
        password = request.form["password"]
        insert_sql = "INSERT INTO REGISTER VALUES (?,?,?)"
        prep_stmt = ibm_db.prepare(conn, insert_sql)
        ibm_db.bind_param(prep_stmt, 1, name)
        ibm_db.bind_param(prep_stmt, 2, email)
        ibm_db.bind_param(prep_stmt, 3, password)
        ibm_db.execute(prep_stmt)
        msg = "You have successfully registered!"
    return render_template('login.html', msg=msg)
@app.route("/login", methods=['POST', 'GET'])
def login1():
    if request.method == "POST":
        email = request.form["email"]
        password = request.form["password"]
        sql = "SELECT * FROM REGISTER WHERE EMAIL=? AND PASSWORD=?"
        stmt = ibm_db.prepare(conn, sql)
        ibm_db.bind_param(stmt, 1, email)
        ibm_db.bind_param(stmt, 2, password)
        ibm_db.execute(stmt)
        account = ibm_db.fetch_assoc(stmt)
        print(account)
        if account:
            session['Loggedin'] = True
            session['id'] = account['EMAIL']
            session['email'] = account['EMAIL']
            return render_template('model.html')
        else:
            msg = "Incorrect Email/password"
            return render_template('login.html', msg=msg)
    else:
        return render_template('login.html')
@app.route('/modelq')
def liv_pred():
    cap = cv2.VideoCapture('carParkingInput.mp4')
    with open('parkingSlotPosition', 'rb') as f:
        posList = pickle.load(f)
    width, height = 107, 48
    def checkParkingSpace(imgPro):
        spaceCounter = 0
        for pos in posList:
            x, y = pos
            imgCrop = imgPro[y:y + height, x:x + width]
            count = cv2.countNonZero(imgCrop)
            if count < 900:
                color = (0, 255, 0)
                thickness = 3
                spaceCounter += 1
            else:
                color = (0, 0, 255)
                thickness = 2
            cv2.rectangle(img, pos, (pos[0] + width, pos[1] + height), color, thickness)
        cvzone.putTextRect(img, f'Free: {spaceCounter}/{len(posList)}', (100, 50), scale=3, thickness=5, offset=20,
                           colorR=(200, 0, 0))
    while True:

        if cap.get(cv2.CAP_PROP_POS_FRAMES)==cap.get(cv2.CAP_PROP_FRAME_COUNT):
            cap.set(cv2.CAP_PROP_POS_FRAMES, 0)
        success, img = cap.read()
        if not success:
            break
        imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
        imgBlur = cv2.GaussianBlur(imgGray, (3, 3), 1)
        imgThreshold = cv2.adaptiveThreshold(imgBlur, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY_INV,
                                             25, 16)
        #cv2.imshow("image",imgGray)
        imgMedian = cv2.medianBlur(imgThreshold, 5)
        kernel = np.ones((3, 3), np.uint8)
        imgDilate = cv2.dilate(imgMedian, kernel, iterations=1)
        checkParkingSpace(imgDilate)
        cv2.imshow("Image", img)
        if cv2.waitKey(1) & 0xFF == ord('q'):
            break
    cap.release()
    cv2.destroyAllWindows()
    return redirect(url_for('login1'))
if __name__ == "__main__":
    arupp. n(debug=True)
```

Signup.html:

```
<!DOCTYPE html>
<html>
<head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title> Responsive Login and Signup Form </title>
     <link href='https://fonts.googleapis.com/css?family=Montserrat' rel='stylesheet'>
    <link href='https://unpkg.com/boxicons@2.1.2/css/boxicons.min.css' rel='stylesheet'>
    <link href='https://fonts.googleapis.com/css?family=Press Start 2P' rel='stylesheet'>
    <link rel="stylesheet" type="text/css" href=".\static\signup.css" />
</head>
<body>
    <div class="navbar">
        <ul>
            <li><h3>AI &nbspENABLED &nbspCAR &nbspPARKING &nbspUSING &nbspOPENCV</h3></li>
            <div class="navi">
            <li><a class="active" href="/index.html">Home</a></li>
            <li><a href="/login.html">Login</a></li>
            <li><a href="/signup.html">Register</a></li>
        </ul>
    </div>
    <section class="container forms">
        <div class="form login">
            <div class="form-content">
                <header>Create an Account</header>
                <p>Enter your personal details to create account</p>
                <form action="signup" method="post">
                    <div class="field input-field">
                        <label for="name">Name</label><br>
                <input type="name" name="name" placeholder="Enter Name" class="input">
                    </div><br>
                    <div class="field input-field">
                        <label for="email">Email</label><br>
                        <input type="email" placeholder="Enter Email" name="email" class="input">
                    </div><br>
                    <div class="field input-field">
                        <label for="password">Password</label><br>
                        <input type="password" placeholder="create password" name="password" class="password">
                    </div><br>
                    <div class="field button-field">
                        <button>Signup</button>
                    </div>
                </form>
                <div class="form-link">
                    <span>Already have an account? <a href="/login.html" class="link login-link">Login</a></span>
                </div>
```

```
            </div>
          </div>
    </body>
  </html>
```

Index.html:

```
<!DOCTYPE html>
  <html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title>Home</title>
    <link rel="stylesheet" type="text/css" href=".\static\index.css" />
  </head>
  <body>
    <div class="navbar">
      <ul>
        <li><a class="active"
href="file:///C:/Users/rmega/Downloads/Project/flask/uploads/templates/in
dex.html">Home</a></li>
        <li><a href="aboutus.html">About us</a></li>
        <li><a href="login.html">Login</a></li>
        <li><a href="signup.html">Register</a></li>
        <li><a href="contact.html">Contact</a></li>
      </ul>
    </div>
    <br><br><br><br>
    <h1>PARKING DETECTION</h1>
<p><i>AI enable car parking using opencv</i></p><br>
        <p><button class="left" >
        <a href="/login.html">GET STARTED</a>
        </button></p>
  </body>
</html>
```

Login.html:

```
<!DOCTYPE html>
  <html lang="en">
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title> Responsive Login and Signup Form </title>
<linkhref='https://fonts.googleapis.com/css?family=Montserrat'rel='stylesheet'>
<linkhref='https://unpkg.com/boxicons@2.1.2/css/boxicons.min.css'rel='styleshee
t'>
<link href='https://fonts.googleapis.com/css?family=Press Start 2P'
rel='stylesheet'>
    <link rel="stylesheet" type="text/css" href=".\static\login.css" />
  </head>
  <body>
    <div class="navbar">
      <ul>
        <li><h3>AI &nbspENABLED &nbspCAR &nbspPARKING
&nbspUSING &nbspOPENCV</h3></li>
      <div class="navi">
        <li><a class="active" href="/index.html">Home</a></li>
        <li><a href="/login.html">Login</a></li>
        <li><a href="/signup.html">Register</a></li>
      </div>
      </ul>
    </div>
    <section class="container forms">
      <div class="form login">
        <div class="form-content">
          <header>Login to Your Account</header>
          <form action="login" method="post">
            <div class="field input-field">
              <label for="email">Email</label><br>
              <input type="email" placeholder="Email" name="email"
class="input">
            </div><br>

            <div class="field input-field">
              <label for="password">Password</label><br>
              <input type="password" placeholder="Password"
name="password" class="password">
            </div><br>
            <div class="field button-field">
              <button>Login</button>
            </div>
          </form>
          <div class="form-link">
            <span>Don't have an account? <a href="/signup.html" class="link
signup-link">Signup</a></span>
          </div>
        </div>
      </div>
      </section>
  </body>
</html>
```

Model.html:

```
<!DOCTYPE html>
  <html>
  <head>
    <meta charset="UTF-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <meta name="viewport" content="width=device-width, initial-
scale=1.0">
    <title> Responsive Login and Signup Form </title>
    <link href='https://fonts.googleapis.com/css?family=Montserrat'
rel='stylesheet'>
    <link
href='https://unpkg.com/boxicons@2.1.2/css/boxicons.min.css'
rel='stylesheet'>
    <link href='https://fonts.googleapis.com/css?family=Press Start
2P' rel='stylesheet'>
  <link rel="stylesheet" type="text/css" href=".\static\model.css" />
  </head>
  <body>
    <div class="navbar">
      <ul style="padding-left: 1150px;">
        <li><a href="/index.html">Logout</a></li>
      </ul>
    </div>
    <section class="container forms">
      <div class="form login">
        <div class="form-content">
          <header>Check the Parking Slot</header>
          <form action="modelq">
            <div class="field button-field">
              <button>👉Click here</button>
            </div>
          </form>
        </div>
      </div>
    </section>
  </body>
</html>
```

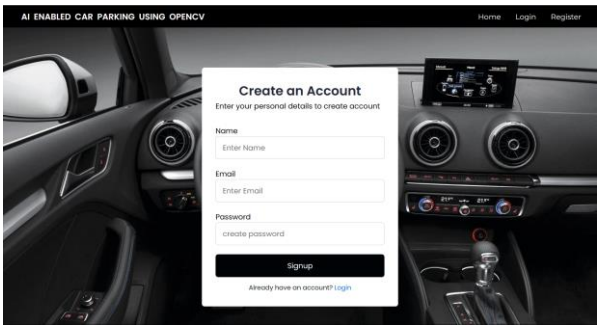
Aboutus.html:

```
<!DOCTYPE html>
  <html lang="en">
  <head>
    <title> Responsive Login and Signup Form </title>
    <linkhref='https://fonts.googleapis.com/css?family=Montserrat'
rel='stylesheet'>
    <link
href='https://unpkg.com/boxicons@2.1.2/css/boxicons.min.css'
rel='stylesheet'>
    <link href='https://fonts.googleapis.com/css?family=Press Start
2P' rel='stylesheet'>
    <link rel="stylesheet" type="text/css"
href=".\static\aboutcss.css" />
  </head>
  <body><div class="navbar"> <ul>
        <li><a class="active" href="/index.html">Home</a></li>
        <li><a href="/aboutus.html">About us</a></li>
        <li><a href="/login.html">Login</a></li>
        <li><a href="/signup.html">Register</a></li>
        <li><a href="#contact">Contact</a></li>
        </ul></div>
<br><br><br><header>About</header>
    <br><br><img
src="https://www.facilitiesnet.com/resources/editorial/2019/parking-
garage-43252.jpg" >
<h2>Al Enable car parking using OpenCV</h2>
<p>The Al-Enabled Car Parking System Utilizing OpenCV
Technology is a cutting-edge project that aims to revolutionize the
way parking lots operate. This system uses OpenCV, a popular
computer vision library, to enable vehicles to park
autonomously.</p>
<br><p><span style="color:red">&#10004 </span>The OpenCV
library analyzes the footage and identifies the available parking
spaces in the lot.<br><br>
<span style="color:red">&#10004</span>The system is designed to
be highly accurate, and it can detect small and large vehicles, even in
low-light conditions.</p>
</body>
</html>
```

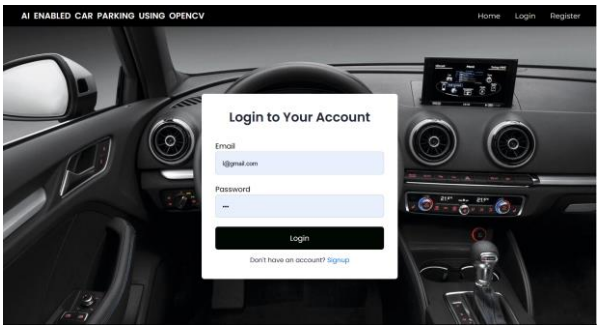
OUTPUT:

Homepage:

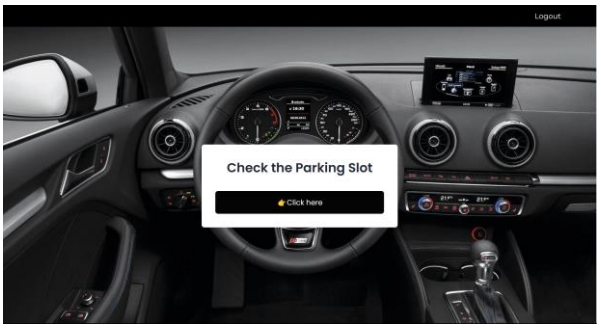
Register page:



Login Page:



Checking Page:



Model Page: